

07/23/99
15490 U.S. PTO

pbs

Please type a plus sign (+) inside this box → ☐

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.


PTO/SB/05 (1/98)
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
OMB 0651-0032

UTILITY PATENT APPLICATION TRANSMITTAL (Only for new nonprovisional applications under 37 CFR 1.53(b))	Attorney Docket No.	X-528 US
	First Named Inventor or Appl. Identifier	Reto Stamm
	Title	System and Method for Testing Parameterized Logic Cores
	Express Mail Label No.	EM253371423US

APPLICATION ELEMENTS See MPEP chapter 600 concerning utility patent application contents.	ADDRESS TO: Assistant Commissioner for Patents Box Patent Application Washington, DC 20231
1. <input checked="" type="checkbox"/> Fee Transmittal Form (e.g., PTO/SB/17) (Submit an original, and a duplicate for fee processing)	6. <input type="checkbox"/> Microfiche Computer Program (Appendix)
2. <input checked="" type="checkbox"/> Specification [Total Pages 17] (preferred arrangement set forth below) <ul style="list-style-type: none">- Descriptive title of the Invention- Cross References to Related Applications- Statement Regarding Fed sponsored R & D- Reference to Microfiche Appendix- Background of the Invention- Brief Summary of the Invention- Brief Description of the Drawings (if filed)- Detailed Description- Claim(s)- Abstract of the Disclosure	7. <input type="checkbox"/> Nucleotide and/or Amino Acid Sequence Submission (if applicable, all necessary) <ul style="list-style-type: none">a. <input type="checkbox"/> Computer Readable Copyb. <input type="checkbox"/> Paper Copy (identical to computer copy)c. <input type="checkbox"/> Statement verifying identity of above copies
3. <input checked="" type="checkbox"/> Drawing(s) (35 USC 113) [Total Sheets 7]	ACCOMPANYING APPLICATION PARTS 8. <input type="checkbox"/> Assignment Papers (cover sheet & document(s)) 9. <input type="checkbox"/> 37 CFR 3.73(b) Statement (when there is an assignee) <input type="checkbox"/> Power of Attorney 10. <input type="checkbox"/> English Translation Document (if applicable) 11. <input type="checkbox"/> Information Disclosure Statement (IDS)/PTO-1449 <input type="checkbox"/> Copies of IDS Citations 12. <input type="checkbox"/> Preliminary Amendment 13. <input checked="" type="checkbox"/> Return Receipt Postcard (MPEP 503) (Should be specifically itemized) 14. <input type="checkbox"/> *Small Entity Statement(s) <input type="checkbox"/> Statement filed in prior application, Status still proper and desired (PTO/SB/09-12) 15. <input type="checkbox"/> Certified Copy of Priority Document(s) (if foreign priority is claimed) 16. <input type="checkbox"/> Other:
4. Oath or Declaration [Total Pages 2] <ul style="list-style-type: none">a. <input type="checkbox"/> Newly executed (original or copy)b. <input type="checkbox"/> Copy from a prior application (37 CFR 1.63(d)) (for continuation/divisional with Box 17 completed)<ul style="list-style-type: none">[Note Box 5 below]i. <input type="checkbox"/> DELETION OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 CFR 1.63(d)(2) and 1.33(b).	
5. <input type="checkbox"/> Incorporation By Reference (useable if Box 4b is checked) The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b, is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.	
17. If a CONTINUING APPLICATION , check appropriate box and supply the requisite information below and in a preliminary amendment <input type="checkbox"/> Continuation <input type="checkbox"/> Divisional <input type="checkbox"/> Continuation-in-part (CIP) of prior application No: _____ Prior application information: Examiner _____ Group / Art Unit: _____	

18. CORRESPONDENCE ADDRESS	
<input type="checkbox"/> Customer Number or Bar Code Label	(Insert Customer No. or Attach bar code label here) or <input checked="" type="checkbox"/> Correspondence address below

Name	Jeanette S. Harms				
	XILINX, INC.				
Address	2100 Logic Drive				
City	San Jose	State	California	Zip Code	95124
Country	US	Telephone	408-879-4969	Fax	408-377-6137

Name (print/Type)	Jeanette S. Harms	Registration No. (Attorney/Agent)	35,537
Signature		Date	July 23, 1999

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231.

1 SYSTEM AND METHOD FOR TESTING
2 PARAMETERIZED LOGIC CORES

3 Reto Stamm
4 Mary O'Connor
5 Christophe Brotelande
6

7 FIELD OF THE INVENTION

8 The present invention generally relates to testing
9 parameterized logic cores, and more particularly to
10 improving test coverage of parameterized logic cores.
11

12 BACKGROUND

13 Logic cores are becoming increasingly popular in the
14 design of integrated circuits. Whereas in the past,
15 integrated circuits were largely designed from scratch,
16 today's integrated circuits are designed using pre-designed
17 logic elements that are made available in logic cores. One
18 reason for the popularity of logic cores is the trend toward
19 standards-based implementations. Additional reasons
20 includes desires to speed the design process and reduce
21 costs.

22 Parameterized logic cores permit a designer to
23 partially customize predefined logic circuits. For example,
24 a synchronous FIFO element of a logic core may include port-
25 width and depth parameters which allow the designer to
26 tailor the logic to satisfy application-specific
27 requirements.

28 Designers who use logic cores rely on the fact that the
29 elements operate correctly, both individually and when
30 connected to other logic elements. Thus, logic core vendors
31 must sufficiently test the logic, not only to verify that
32 the circuit performs the specified function for commonly
33 used parameters, but also to verify that circuit operates
34 satisfactorily for infrequently used parameters.

35 While testing individual elements of a logic core using
36 various permutations of the available parameters may be
37 feasible, the number of permutations of parameters when

1 considering various combinations of the logic elements in
2 the core quickly escalates. Thus, there may be more
3 permutations or parameters for a logic core than can
4 feasibly be tested.

5 Testers often resort to constructing test sets that
6 reflect a representative set of possible parameter
7 permutations. While this approach can be tailored to fit
8 the time available for testing, the associated risk is that
9 some errors may go undetected. To increase test coverage
10 more resources must be devoted to testing.

11 A method that address the aforementioned problems, as
12 well as other related problems, is therefore desirable.

13

14 SUMMARY OF THE INVENTION

15 In various embodiments, the invention provides a
16 computer-implemented method and system for testing a
17 parameterizable logic core. A set of parameter values for
18 the logic core is randomly generated. Using the random
19 parameter values, a netlist is generated, and circuit
20 behavior is simulated.

21 In one embodiment, the random parameter values are
22 provided as input to a graphical user interface (GUI) of a
23 core generator. The capability of the GUI to respond to
24 erroneous parameter sets is thereby tested.

25 In another embodiment, if simulation of circuit
26 behavior in association with a particular parameter set
27 results in an error, the parameter set is cloned and mutated
28 to generate a netlist for another simulation. Cloning and
29 mutating the parameter sets can assist in replicating error
30 conditions.

31 The above summary of the present invention is not
32 intended to describe each disclosed embodiment of the
33 present invention. The figures and detailed description
34 that follow provide additional example embodiments and
35 aspects of the present invention.

36

1 BRIEF DESCRIPTION OF THE DRAWINGS

2 Various aspects and advantages of the invention will
3 become apparent upon review of the following detailed
4 description and upon reference to the drawings in which:

5 FIG. 1 is a flowchart of a process for testing
6 parameterized logic cores in accordance with an example
7 embodiment of the invention;

8 FIG. 2 is a block diagram of a first example system for
9 testing parameterized logic cores;

10 FIG. 3 is a block diagram of a second example system
11 for testing parameterized logic cores;

12 FIG. 4 is a flowchart of a process for randomly
13 generating parameters using an example formula-based
14 approach;

15 FIG. 5 is a flowchart of a process for randomly
16 generating parameters in a GUI-based system;

17 FIGS. 6A, 6B, and 6C show example GUI screens to which
18 random parameters are input and then displayed; and

19 FIG. 7 is a flowchart of an example process for
20 mutating parameters of a parameter set.

21 While the invention is susceptible to various
22 modifications and alternative forms, specific embodiments
23 thereof have been shown by way of example in the drawings
24 and will herein be described in detail. It should be
25 understood, however, that the detailed description is not
26 intended to limit the invention to the particular forms
27 disclosed. On the contrary, the intention is to cover all
28 modifications, equivalents, and alternatives falling within
29 the spirit and scope of the invention as defined by the
30 appended claims.

31

32 DETAILED DESCRIPTION

33 The present invention is believed to be applicable to a
34 variety of methods and systems for testing parameterizable
35 logic cores. The present invention has been found to be
36 particularly applicable and beneficial in testing

1 parameterizable logic cores for PLDs. While the present
2 invention is not so limited, an appreciation of the present
3 invention is presented by way of specific examples involving
4 PLDs.

5 FIG. 1 is a flowchart of a process for testing
6 parameterized logic cores in accordance with an example
7 embodiment of the invention. The process generally
8 comprises creating a random set of parameters for the logic
9 core, testing the parameterized logic core, saving test
10 information for further analysis, and creating further sets
11 of random parameters for additional testing. The process of
12 automatically, randomly creating parameter sets is believed
13 to provide enhanced coverage relative to all possible
14 parameter permutations. In addition, evolving parameter
15 sets in which core testing failed is designed to help
16 isolate errors in the core.

17 At step 102 a set of random parameters is created. The
18 set of parameters may be provided either directly to core
19 generator software or as input to a GUI interface to a core
20 generator. FIG. 4 illustrates a script-based process for
21 generating random parameters, and FIG. 5 illustrates a GUI-
22 based process for generating random parameters. In either
23 instance, either the core generator itself or the GUI must
24 check for parameter values that are beyond permitted limits.

25 The random parameter set is created using conventional
26 methods for generating random numbers. The upper and lower
27 limits used in generating the random numbers can be, for
28 example, reasonable levels beyond what is permitted for the
29 core (to test the core generator's or GUI's handling of
30 invalid parameters).

31 In another embodiment, the random parameters are
32 generated using selected, respective probabilities to weight
33 the generation of random parameters. Thus, a parameter
34 value having a greater weight would be more likely to be
35 generated than would a parameter value having a lower
36 weight. The probabilities may be assigned in two ways, for

1 example. The first method for assigning probabilities is to
2 specify specific parameter values as having higher
3 probabilities. For example, if a parameter has a range of
4 values from 1 to 8, it may be desirable to run more tests
5 using parameter values of 1 and 8 than tests where the
6 parameter value is out of range.

7 The second method assigns probabilities based on past
8 test performance. For example, if a parameterized logic
9 core fails a test, then the parameter values used in
10 generating the logic core are assigned increased
11 probabilities. Thus, subsequent tests may exhibit a
12 tendency toward emphasizing parameter values causing tests
13 to fail. This permits faster recognition of particular
14 parameter values that cause the test to fail.

15 An assigned probability may be strictly associated with
16 a discrete value. For example, the probability .75 may be
17 associated with the parameter value 16. Alternatively, a
18 probability may be applied to a selected value and to values
19 near the selected value. For example, a selected parameter
20 value 24 may be assigned a probability .40, parameter values
21 23 and 25 assigned parameter values .20, and the remaining
22 parameter values having equal probabilities.

23 At step 106, a netlist is created from the selected
24 core and random parameter set. Various core generator tools
25 are available for such purpose. For example, the Xilinx
26 CORE Generator™ System can be used to generate a netlist
27 for XILINX programmable logic devices. It will be
28 appreciated that other device-specific tools are available
29 from other vendors.

30 At step 108 using conventional methods, the netlist is
31 integrated with a test bench, and behavior of the
32 parameterized logic core is simulated using conventional
33 simulation tools. If no errors are detected in the
34 simulation, decision step 110 directs control to decision
35 step 112. If there are further implementation steps, for
36 example, mapping to a particular FPGA device or place-and-

1 route, then control is directed to step 114 to run the next
2 implementation tool on the design. The process of steps
3 108-114 is repeated until there is a simulation failure or
4 there are no more implementation steps to perform.

5 At step 116, the parameter set, the test results (e.g.,
6 pass or fail), and the level of testing completed are stored
7 for future reference. Stored in addition are version levels
8 of core generator software and devices on which the cores
9 were simulated. The information is collected in a
10 relational database system, for example a database system
11 from Oracle, which is useful for analyzing results of
12 thousands of tests.

13 If a test fails, decision step 118 directs control to
14 step 120, where the parameter set is cloned a selected
15 number of times. The number of clones may be, for example,
16 a function of the number of possible permutations, the
17 number of tests already performed on the core, the number of
18 generations parameter sets already cloned for the core, and
19 associated factors used to scale the other variables.

20 Each of the cloned parameter sets is then mutated at
21 step 122. In an example embodiment, the number of
22 parameters to mutate decreases with the number of
23 generations of parameter sets tested, and the particular
24 parameters that are mutated are randomly selected, for
25 example. The number of generations of parameter sets tested
26 depends on the time available for testing and the number of
27 parameters.

28 At step 124, the processing set forth in steps 106-122
29 is repeated for each of the mutated parameter sets. It will
30 be appreciated that, while not shown, a parameter set that
31 is invalid need not be used in testing the core.

32 The process continues at step 126, where steps 102-124
33 are repeated, beginning with a random parameter set that is
34 newly created at step 102. The number of iterations may
35 depend on a variety of factors, such as the number of
36 parameters, the time available, and the number failed tests.

1 It will be appreciated that a large number of failures
2 indicates that running additional tests may be a waste of
3 resources, while no failures indicates that additional tests
4 may be useful in uncovering errors.

5 When testing is complete, step 128 reports parameter
6 value distribution and test failures per parameter value.
7 For example, for parameters A, B, and C, the report may
8 include the following information:

9 A: 2(4)[2] 3(8) **4(1)[1]** 6(3)

10 B: true(13) **false(3)[3]**

11 C: Octal(8)[1] Hexadecimal(4)[1] Decimal(4)[1]

12 The example format for the report is a list for each
13 parameter. Each list includes the tested parameter values
14 and associated numbers of tests and test failures. The
15 number in parentheses is the total number of tests performed
16 for that parameter value, and the number in square brackets
17 indicated the number of tests failed for the parameter
18 value. For example, parameter A was assigned a value of 2
19 for a total of 4 tests, and two of the tests failed;
20 parameter A was assigned a value of 3 for 8 tests, and all 8
21 tests passed; and parameter A was assigned a value of 4 for
22 1 of the tests and the one test failed.

23 Selected parameter values are highlighted to assist in
24 debugging the tested core. For example, if every test fails
25 when a parameter has a certain value, then this information
26 may be helpful in isolating a problem in the core.
27 Therefore, the example parameter report highlights a
28 parameter value where every test fails when the parameter
29 has that value. In the example report above, the value 4
30 for parameter A and the value "false" for parameter B is
31 highlighted.

32 FIG. 2 is a block diagram of a first example system 160
33 for testing parameterized logic cores. System 160 includes
34 test controller 162, core generator 164, and simulator 166.
35 Test controller 162, core generator 164, and simulator 166
36 are all software tools operable on various data processing

1 systems running various versions of Unix and Windows
2 operating systems, for example.

3 In one embodiment, test controller 162 is a script that
4 interfaces with both core generator 164 and simulator 166.
5 Random parameter sets 168 are created by test controller 162
6 in accordance with the methods described in FIG. 1.

7 Responsive to test controller 162, core generator 164
8 reads a random parameter set and a selected logic core 170,
9 and generates netlist 172. An example core generator is the
10 CORE Generator software system from XILINX. Netlist 172 is
11 then combined with test bench 174 by simulator 166. Example
12 tools that are suitable for simulator 166 include VHDL and
13 VERILOG simulators. Those skilled in the art will recognize
14 other suitable core generators and simulators that could be
15 used in various embodiments of the invention.

16 Test controller 162 receives test result data from
17 simulator 166 and stores test data 176. Test data 176
18 includes, for example, whether the test passed or failed,
19 the parameters set, and various other information as
20 described above. Test data 176 is stored on a test-by-test
21 basis to enable subsequent test analysis based on various
22 parameter values.

23 FIG. 3 is a block diagram of a second example system
24 180 for testing parameterized logic cores. System 180
25 includes core generator 182 having a conventional graphical
26 user interface (GUI) 184, and test controller 186 that has
27 GUI driver 188. GUI 184 allows a user to select a
28 particular core 170, enter parameters, and generate netlist
29 190. In a testing core generator 182 and GUI 184, inputs to
30 and outputs from GUI 184 are processed by GUI driver 188.
31 Various tools such as JavaStar from SunTest or WinRunner
32 from Mercury, are available for building drivers to interact
33 with a GUI.

34 Based on parameters indicated by GUI 184 as being
35 available for editing, test controller 186 generates random
36 parameter set 168 and provides the parameters as input to

1 GUI 184. Programming errors in GUI 184 may be detected by
2 test controller 186 where GUI 184 is expected to detect
3 certain input values as errors and does not provide an error
4 indicator as output to GUI driver 188. Thus, test
5 controller 186 can be programmed to test a core 170, core
6 generator 182, and GUI 184.

7 FIG. 4 is a flowchart of a process for randomly
8 generating parameters using an example formula-based
9 approach. The formula-based approach may be used, for
10 example, in system 160 of FIG. 2. The example process
11 generally entails using minimum and maximum values and
12 criteria that are associated with the respective parameters
13 and generating random parameter values that fall within the
14 specified range and meet the specified criteria. The
15 process is said to be "formula-based" because the criteria
16 are generally stated in terms of a Boolean expression.

17 At step 202, a parameter is obtained for the
18 parameterized core, and at step 204, the minimum and maximum
19 values and the criteria associated with the parameter are
20 obtained. In an example embodiment, the minimum and maximum
21 values and associated criteria are stored in a data file.

22 A random number within the specified limits is
23 generated using conventional methods at step 206. If the
24 number does not meet the specified criteria, then decision
25 step 208 returns control to step 206. An example
26 specification of criteria for a parameter set having two
27 parameters, A and B, could be, " $B < A$." Thus, if B is the
28 parameter in process and the parameter value for A has
29 already been created, then the random number generated for B
30 must be less than the parameter value for A.

31 At step 210, the random number is translated from a
32 decimal number to the format, for example, hexadecimal or
33 binary, required by the core generator tool. Decision step
34 212 returns control to step 202 as long as there are more
35 parameters to process. Otherwise, control is returned to
36 the process for testing the parameter set (FIG. 1, 102).

1 FIG. 5 is a flowchart of a process for randomly
2 generating parameters in a GUI-based system 180 (FIG. 3),
3 for example. The process generally entails generating a
4 random value for each editable parameter in a selected core
5 until a valid parameter set is generated.

6 At step 250 a selected core is opened for processing.
7 For example, GUI driver 188 indicates to GUI 184 which core
8 to open. The editable parameters are identified at step
9 252, and a random order in which the parameters are
10 generated and input to the GUI is selected at step 253. A
11 random sequence in which parameters are entered supports
12 testing behavior of the GUI. For example, if values of
13 various ones of the parameters are dependent on one another,
14 and the GUI is programmed to check for legal combinations,
15 then the GUI should detect invalid parameter combinations
16 for any order in which the values are entered.

17 Until there are no more parameters to edit, decision
18 step 254 directs control to step 256.

19 At step 256, a parameter that has not yet been edited
20 is selected, and the lower and upper limits of the parameter
21 are identified at step 258. The lower and upper limits are
22 indicated in interface data provided by GUI 184 to GUI
23 driver 188. At step 260, a random number is generated using
24 the identified limits and selected probabilities. The
25 selected probabilities are those discussed with reference to
26 FIG. 1. Control is returned to decision step 254 after the
27 parameter is updated with the random value.

28 Decision step 254 directs control to step 262 after all
29 the parameters have been updated with random values. If any
30 of the parameters have invalid values, decision step 262
31 directs control to step 264 where an invalid parameter is
32 selected. Steps 266 and 268 repeat the processing of steps
33 258 and 260, respectively, in generating a new random value
34 for the parameter having the invalid value.

35 Once all the parameters have valid values, decision
36 step 262 directs control to decision step 270, where the

1 entire set is compared to parameter sets previously
2 generated. If the new parameter set is a duplicate, it is
3 discarded, and control is directed to step 252 to create a
4 new random parameter set. Otherwise, the parameter set is
5 saved at step 272, and control is returned to continue the
6 test process.

7 FIGs. 6A, 6B, and 6C show example GUI screens to which
8 random parameters are input and then displayed. FIG. 6A is
9 a data entry screen for a Synchronous FIFO core. The
10 following parameters have been entered: "fifo" as the
11 component name, -5 as the Port Width, 48 as the Depth, and
12 Single Port RAM as the Internal Memory type. Because the
13 Port Width range is 4-80, the specified -5 parameter is
14 invalid. The invalid parameter is highlighted, and the
15 dialog box of FIG. 6B indicates the bad value.

16 For the Filter screen of FIG. 6C, "pdafir" is the
17 Component Name, 8 is the Input Width, Signed Input Data is
18 selected, "Symmetry" is selected for Impulse Response, the
19 Number of Taps is 8, and the Coefficient Width is 8. It
20 will be appreciated that the Sign and Impulse Response
21 parameters may be viewed as binary values for the purpose
22 of generating random parameters.

23 FIG. 7 is a flowchart of an example process for
24 mutating parameters of a parameter set. At step 302, the
25 number of parameters to mutate is selected. The number of
26 parameters is selected based on the number of generations of
27 parameter sets already generated and the time available for
28 testing, for example. One parameter in the set is randomly
29 selected at step 304, and the lower and upper limits are
30 identified at step 306. Step 308 generates a random number
31 within the identified limits and using selected
32 probabilities, as explained above. The random number is
33 then stored as the new parameter. While not shown, it will
34 be appreciated that if the value is invalid for the

1 parameter, the process of step 308 is repeated until a valid
2 parameter is generated. Decision step 310 returns control
3 to step 302 as long as there are more parameters to mutate.
4 Otherwise, control is returned to step 120 of FIG. 1.

5 Accordingly, the present invention provides, among
6 other aspects, a system and method for testing parameterized
7 logic cores. Other aspects and embodiments of the present
8 invention will be apparent to those skilled in the art from
9 consideration of the specification and practice of the
10 invention disclosed herein. It is intended that the
11 specification and illustrated embodiments be considered as
12 examples only, with a true scope and spirit of the invention
13 being indicated by the following claims.

14

1 **CLAIMS**

2 What is claimed is:

3 1. A computer-implemented method for testing a
4 parameterizable logic core, comprising:
5 randomly generating a set of parameter values for the
6 logic core;
7 generating a netlist from the set of parameter values
8 and logic core; and
9 simulating circuit behavior with the netlist.

10
11 2. The method of claim 1, further comprising for each
12 parameter:
13 generating a random parameter value within
14 predetermined upper and lower limits associated with the
15 parameter; and
16 generating a new random parameter value if the random
17 parameter value fails to meet predetermined criteria.

18
19 3. The method of claim 2, further comprising:
20 assigning respective probabilities to one or more
21 numbers between the upper and lower limits for one or more
22 of the parameters; and
23 generating the random parameter value as a function of
24 the probabilities.

25
26 4. The method of claim 3, further comprising:
27 providing the parameter value as input to a graphical
28 user interface;
29 generating random replacement values for invalid
30 parameter values detected by the graphical user interface.

31
32 5. The method of claim 1, further comprising:
33 providing the set of parameter values to a graphical
34 user interface; and
35 identifying invalid parameter values with the graphical
36 user interface.

1

2 6. The method of claim 5, further comprising:
3 generating random replacement parameter values for the
4 invalid parameters; and
5 repeating the steps of providing the replacement values
6 to the graphical user interface, identifying invalid
7 parameter values, and generating random replacements until
8 all the parameter values are valid.

9

10 7. The method of claim 5, further comprising:
11 selecting a random order in which to provide parameter
12 values to the graphical user interface; and
13 providing the parameters one-by-one as input to the
14 graphical user interface.

15

16 8. The method of claim 5, further comprising:
17 cloning the set of parameter values;
18 mutating the set of parameter values, whereby a mutated
19 set of parameter values is produced;
20 generating a new netlist from the mutated set of
21 parameter values and logic core; and
22 simulating circuit behavior with the new netlist.

23

24 9. The method of claim 8, wherein the set of parameter
25 values is cloned and mutated only if an error is detected in
26 simulating the circuit behavior.

27

28 10. The method of claim 1, further comprising:
29 cloning the set of parameter values;
30 mutating the set of parameter values, whereby a mutated
31 set of parameter values is produced;
32 generating a new netlist from the mutated set of
33 parameter values and logic core; and
34 simulating circuit behavior with the new netlist.

35

1 11. The method of claim 10, wherein the set of parameter
2 values is cloned and mutated only if an error is detected in
3 simulating the circuit behavior.

4

5 12. The method of claim 11, further comprising repetitively
6 cloning and mutating sets of parameters when errors are
7 detected in simulating the circuit behavior, whereby
8 multiple generations of sets of parameters are created.

9

10 13. The method of claim 12, wherein one or more of the
11 parameter values in a parameter set are mutated.

12

13 14. The method of claim 13, wherein a number of parameter
14 values mutated in a set of parameters is a function of a
15 number of generations previously created.

16

17 15. The method of claim 1, further comprising:
18 identifying which parameters of a graphical user
19 interface are editable; and
20 providing the set of parameter values to the graphical
21 user interface.

22

23 16. The method of claim 15, further comprising:
24 selecting a random order in which to provide the
25 parameter values to the graphical user interface; and
26 providing the values one-by-one as input to the
27 graphical user interface.

28

29 17. The method of claim 1, further comprising:
30 accumulating respective numbers of tests having been
31 performed using different parameter values;
32 accumulating respective numbers of tests failed using
33 each of the parameter values; and
34 highlighting parameters having numbers of failed tests
35 equal to the number of tests.

36

1 18. A system for testing a parameterizable logic core,
2 comprising:
3 a test controller configured and arranged to randomly
4 generate a set of parameter values for the logic core;
5 a core generator coupled to the test controller, the
6 core generator configured and arranged to generate a netlist
7 from the logic core and set of parameter values; and
8 a simulator coupled to the test controller, the
9 simulator configured and arranged to simulating circuit
10 behavior with the netlist and a predetermined test bench.

11
12 19. A system for testing a parameterizable logic core,
13 comprising:
14 a test controller including a GUI-driver, the GUI-
15 driver configured and arranged to randomly generate a set of
16 parameter values for the logic core;
17 a core generator including a GUI coupled to the GUI-
18 driver, the core generator configured and arranged to
19 generate a netlist from the logic core and set of parameter
20 values; and
21 a simulator coupled to the test controller, the
22 simulator configured and arranged to simulate circuit
23 behavior with the netlist and a predetermined test bench.

24
25 20. An apparatus for testing a parameterizable logic core,
26 comprising:
27 means for randomly generating a set of parameter values
28 for the logic core;
29 means for generating a netlist from the set of
30 parameter values and logic core; and
31 means for simulating circuit behavior with the netlist.

1 SYSTEM AND METHOD FOR TESTING
2 PARAMETERIZED LOGIC CORES

3 Reto Stamm
4 Mary O'Connor
5 Christophe Brotelande
6

7
8 ABSTRACT

9 A system and method for testing a parameterizable logic
10 core are provided in various embodiments. A test controller
11 is configured and arranged to generate a set of random
12 parameter values for the logic core. A netlist is created
13 from the parameterized logic core, and circuit behavior is
14 simulated using the netlist. In other embodiments, selected
15 parameter values are optionally weighted to increase the
16 probability of generating those values, and the parameter
17 set is cloned and mutated when simulation fails.

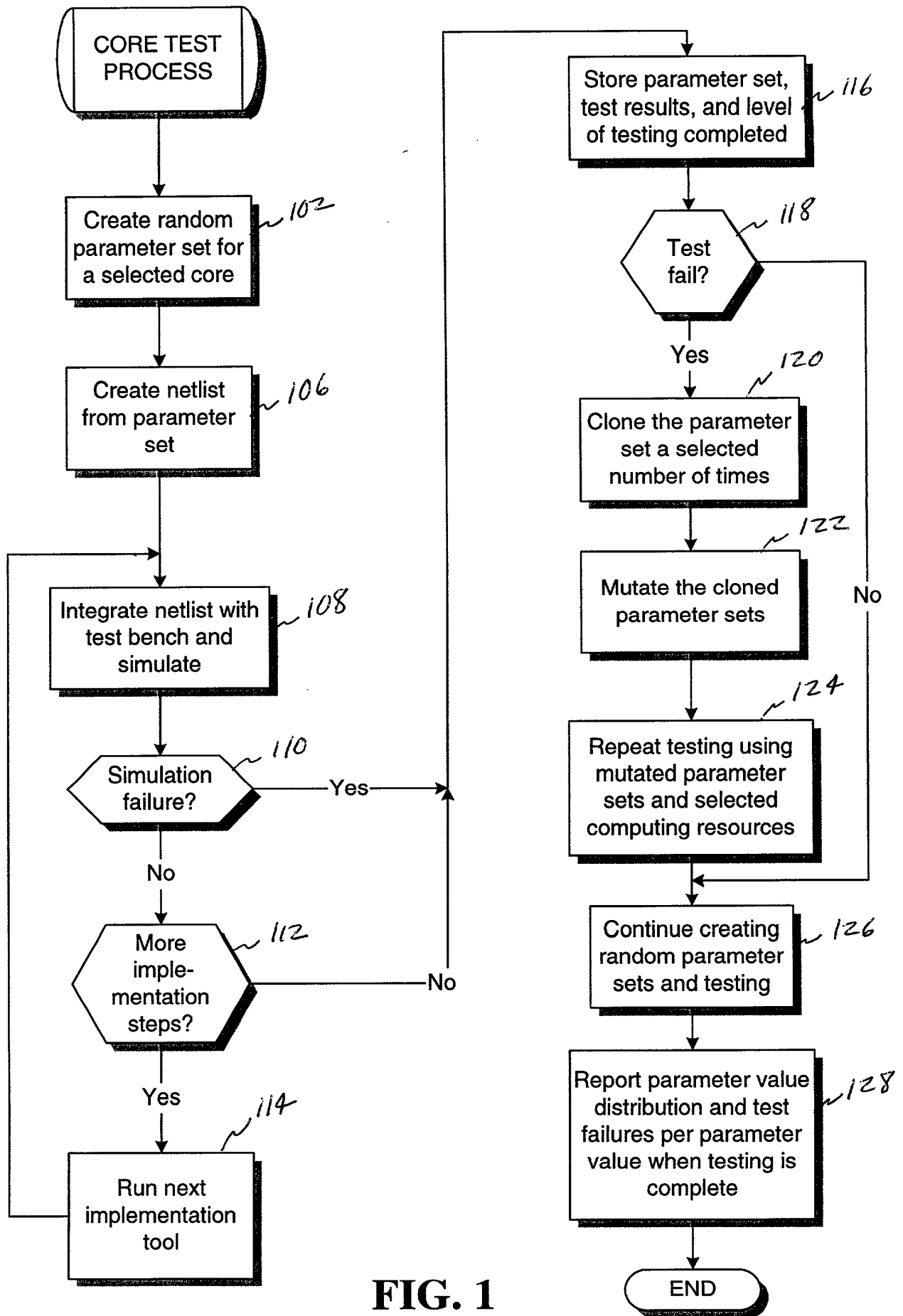


FIG. 1

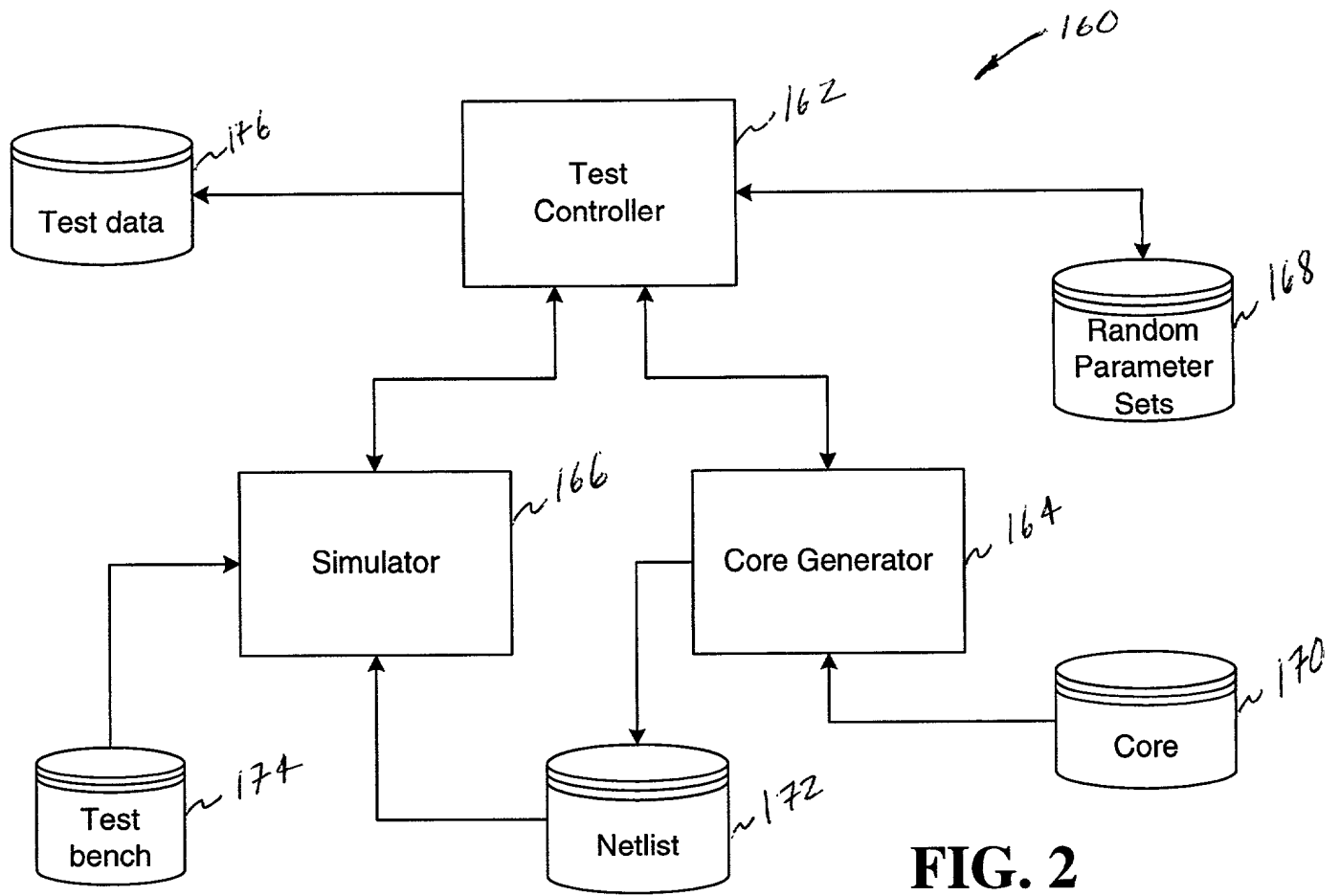


FIG. 2

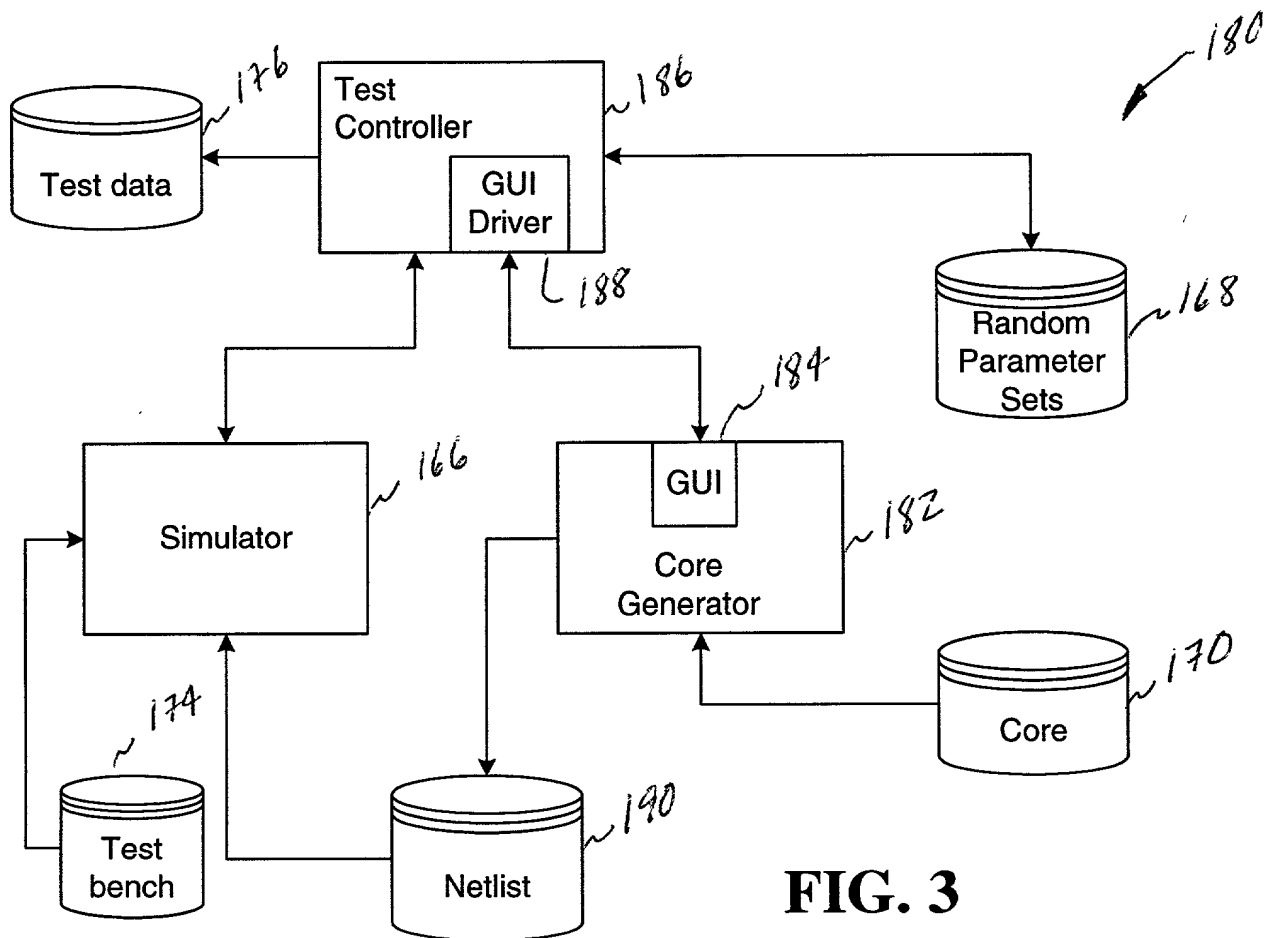


FIG. 3

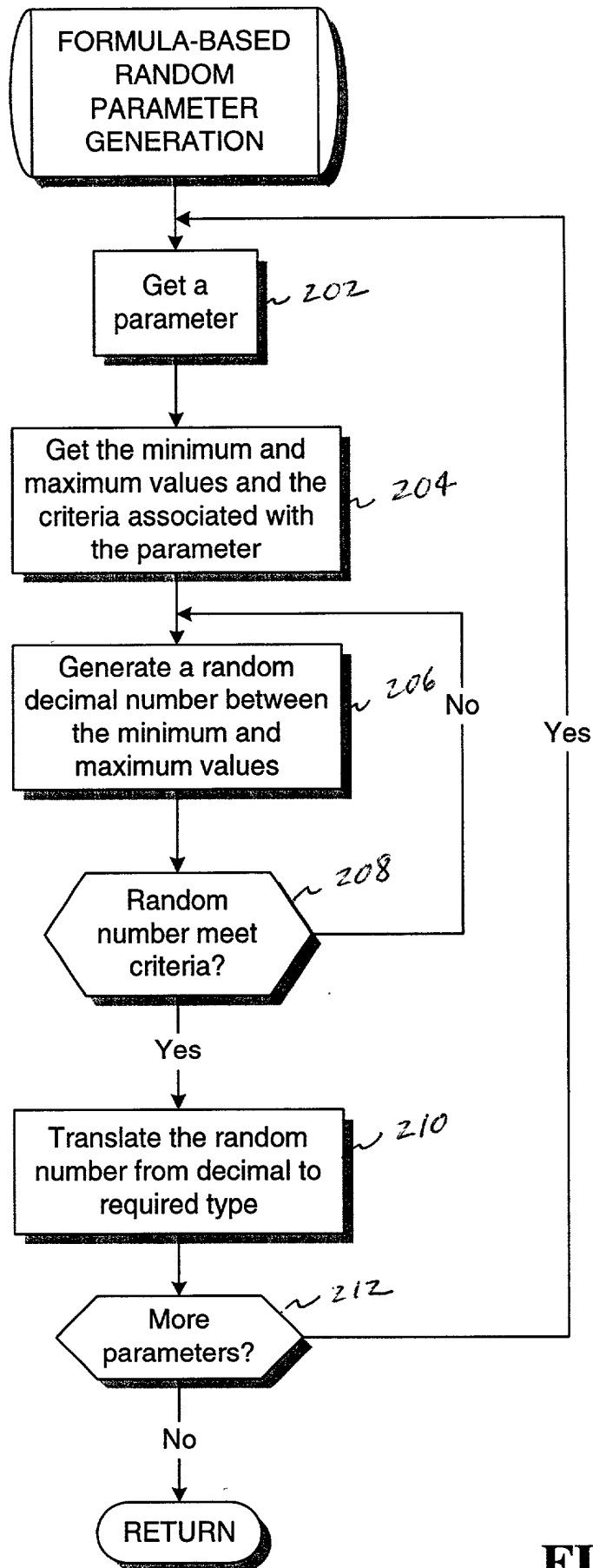


FIG. 4

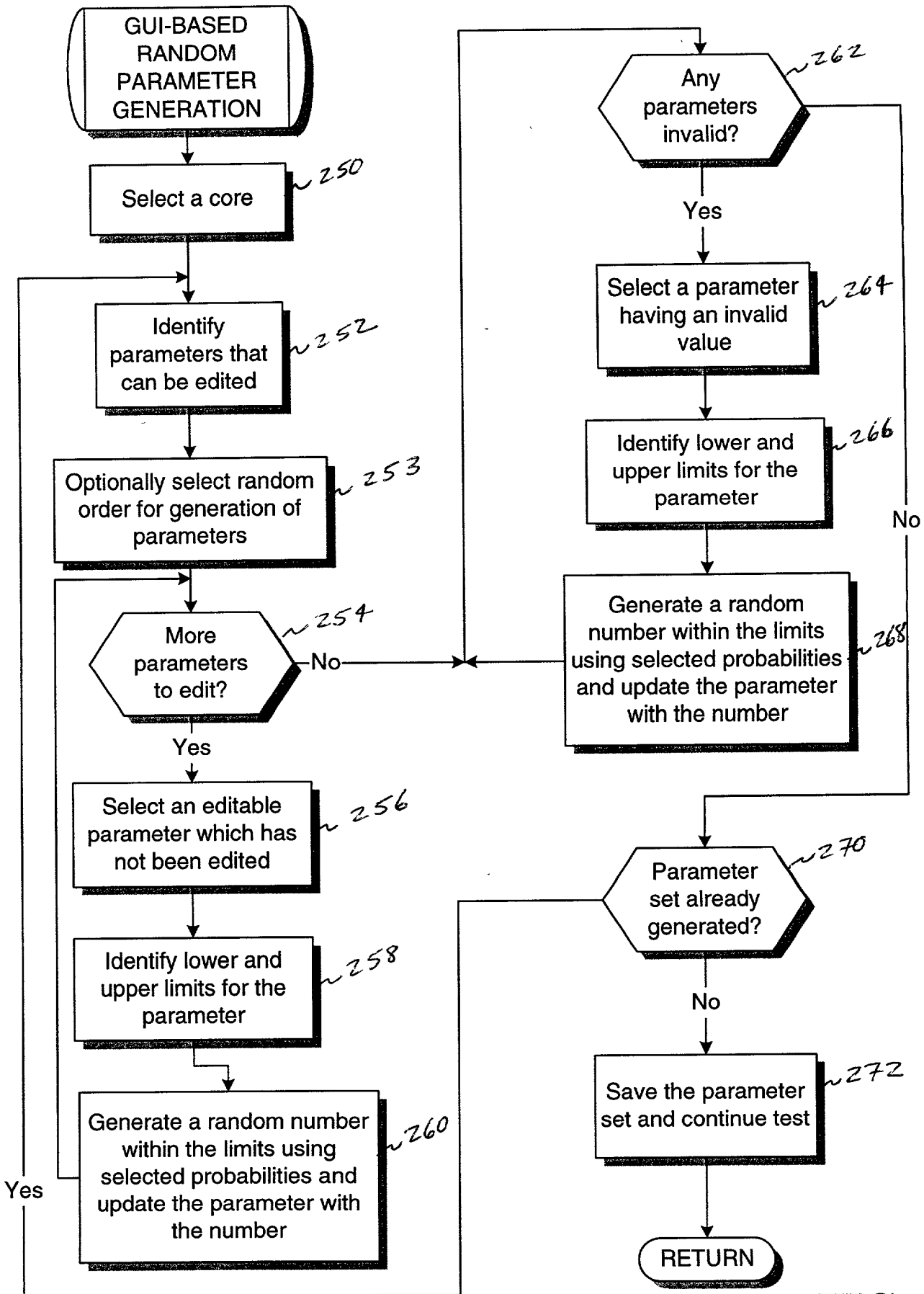


FIG. 5

66620-240560

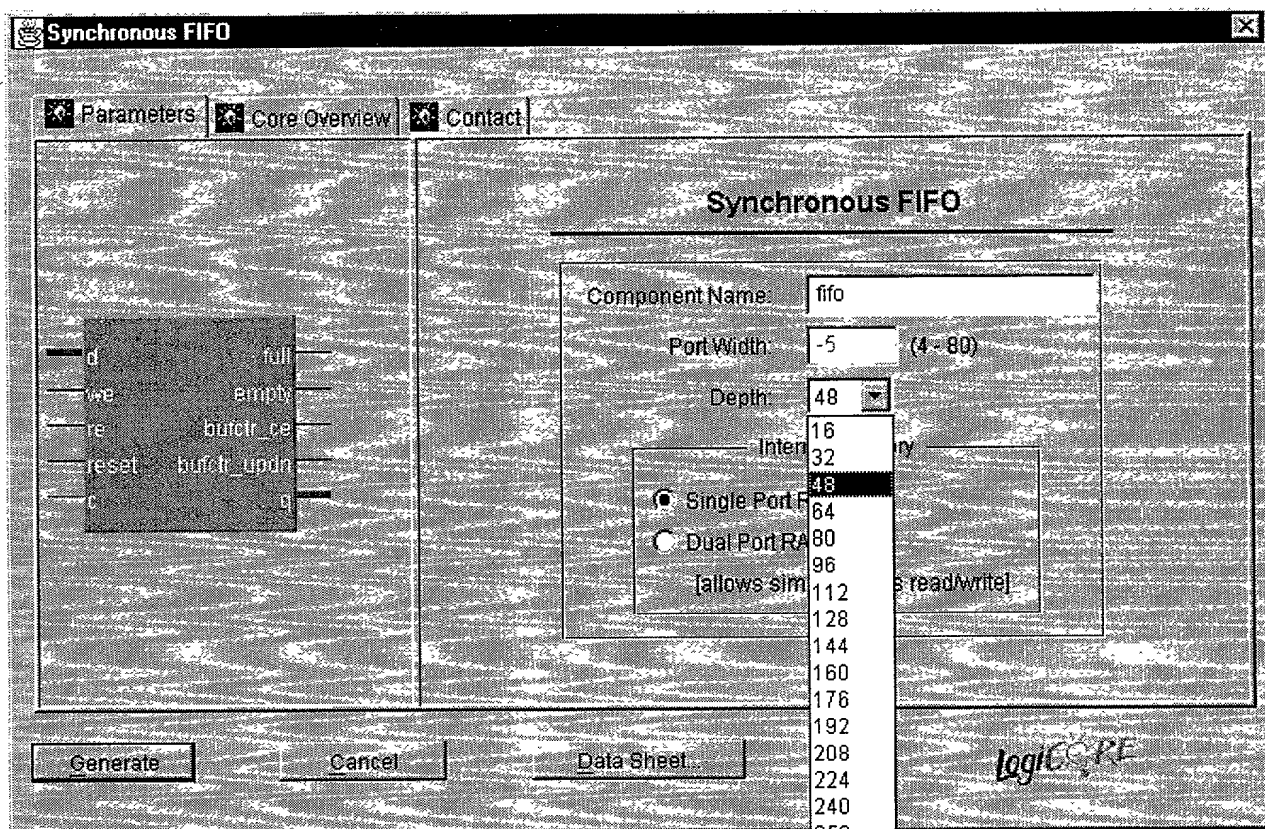


FIG 6A

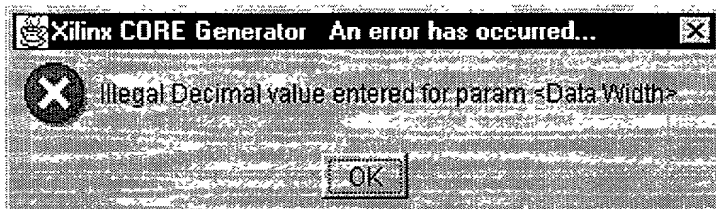


FIG. 6B

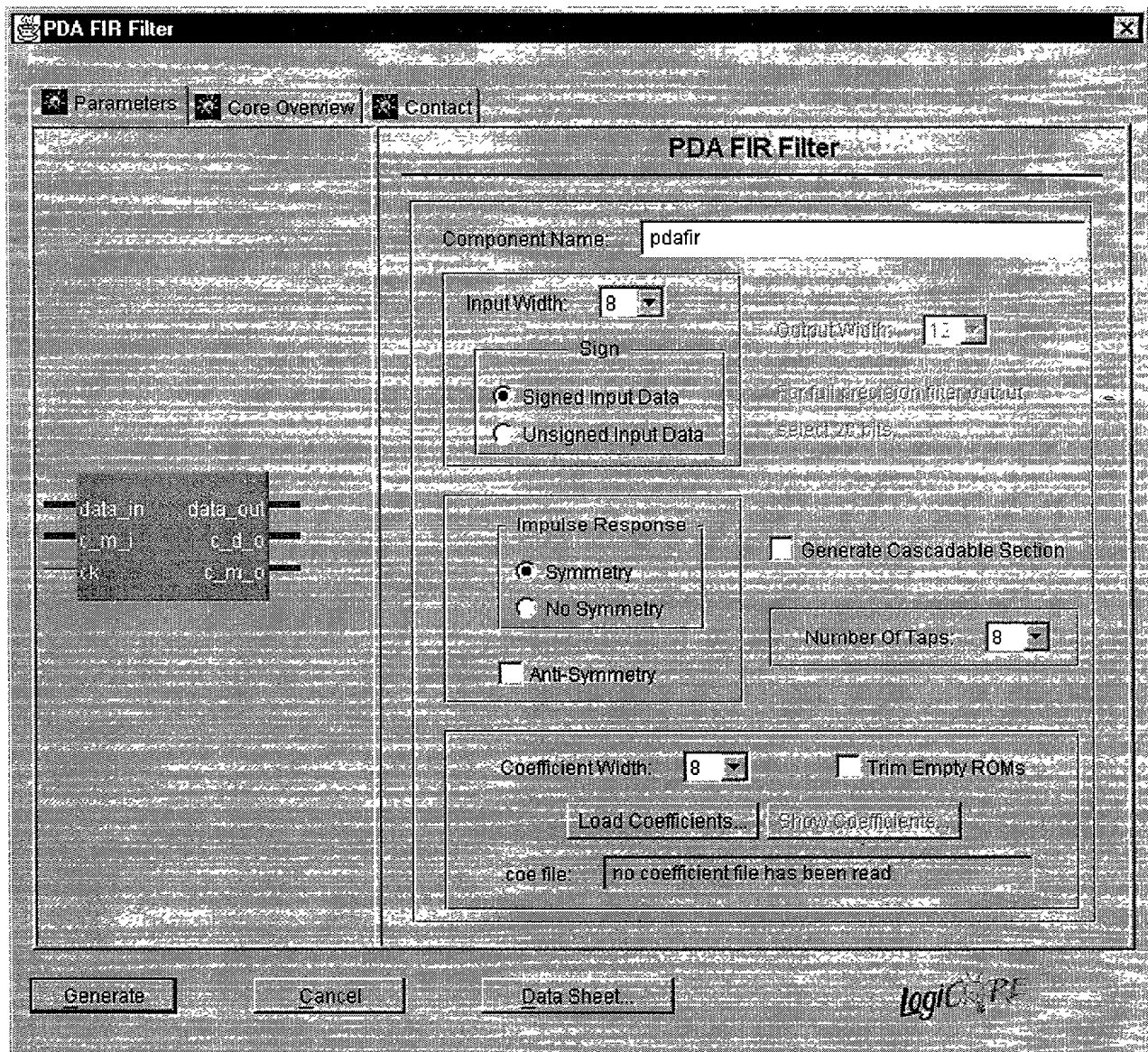


FIG. 6C

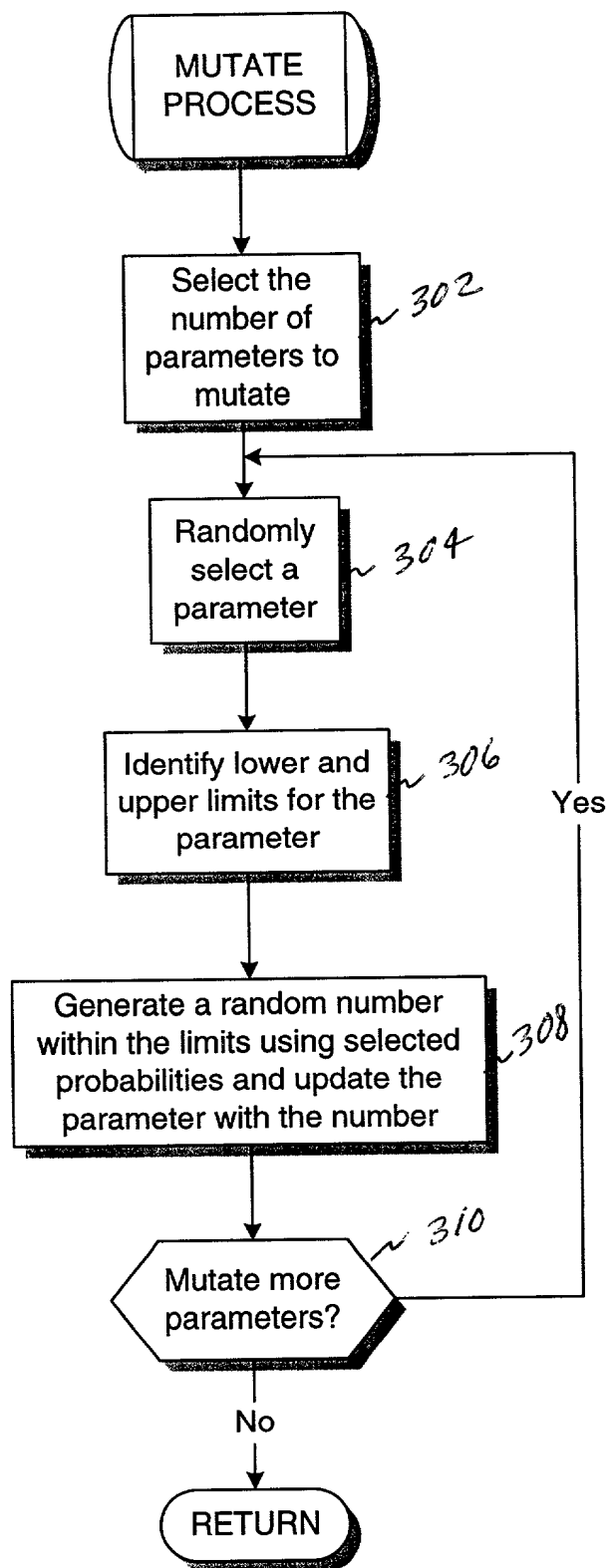


FIG. 7

DECLARATION FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below adjacent to my name.

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of subject matter (process, machine, manufacture, or composition of matter, or an improvement thereof) that is disclosed and/or claimed and for which a patent is solicited by way of the application entitled SYSTEM AND METHOD FOR TESTING PARAMETERIZED LOGIC CORES which (check)

- ☐ is attached hereto.
☐ and is amended by the Preliminary Amendment attached hereto.
☐ was filed on _____ as Application Serial No. _____
☐ and was amended on _____ (if applicable).

I hereby state that I have reviewed and understand the contents of the above identified application, including the claims, including portions amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, § 1.56(a).

I hereby claim foreign priority benefits under title 35, United States Code, § 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s)			Priority Claimed	
			Yes	No
N/A				
(Number)	(Country)	(Day/Month/Year Filed)		
			Yes	No
(Number)	(Country)	(Day/Month/Year Filed)		
			Yes	No
(Number)	(Country)	(Day/Month/Year Filed)		

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) listed below and, insofar as any subject matter of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, § 1.56(a) which occurred between the filing date of the prior application(s) and the national or PCT international filing date of this application:

_____ (Application Serial No.)	_____ (Filing Date)	_____ (Status-patented, pending, abandoned)
_____ (Application Serial No.)	_____ (Filing Date)	_____ (Status-patented, pending, abandoned)

I hereby appoint the following attorney(s) and/or agent(s) to prosecute this application and to transact all business in the United States Patent and Trademark Office connected herewith:

Edel M. Young (32,451), Jeanette S. Harms (35,537), and Lois D. Cartier (40,941) located at Xilinx, Inc., 2100 Logic Drive, San Jose, California 95124.

LeRoy D. Maunu (35,274) located at Crawford PLLC, 333 Washington Avenue N., Suite 5000, Minneapolis, Minnesota 55401.

Address all telephone calls to: Edel M. Young at Telephone No. (408) 879-4969
 Address all correspondence to: Edel M. Young
 Xilinx, Inc.
 2100 Logic Drive
 San Jose, California 95124

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Title 18, United States Code, § 1001 and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or first inventor Reto Stamm

Inventor's signature _____ Date _____
 Residence County Tipperary, Ireland Citizenship Switzerland
 Post Office Address Suirville, Golden, County Tipperary, Ireland

Full name second inventor Mary O'Connor

Inventor's signature _____ Date _____
 Residence Dublin, Ireland Citizenship Ireland
 Post Office Address 8 Arbutus Place, South Circular Road, Dublin, Ireland

Full name third inventor Christophe Brotelande

Inventor's signature _____ Date _____
 Residence Dublin 1, Ireland Citizenship France
 Post Office Address 58b Upper Dorset Street, Apartment 4, Dublin 1, Ireland

Full name fourth inventor _____

Inventor's signature _____ Date _____
 Residence _____ Citizenship _____
 Post Office Address _____

JSH: pbs